Zadání bakalářské práce

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: B 2612 - Elektrotechnika a informatika
Studijní obor: 1802R022 - Informatika a logistika

Analýza možností posuzování spolehlivosti software

Analysis of assessment options of software dependability

Bakalářská práce

Autor: **Adéla Škarydová**
Vedoucí práce: Ing. David Vališ, Ph.D.
Konzultant: doc. RNDr. Miroslav Koucký, CSc.

V Liberci 22. 5. 2009

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Děkuji svému vedoucímu bakalářské práce Ing. Davidu Vališovi, Ph.D. za vstřícnost, trpělivost a cenné rady a připomínky, které mi poskytl při vypracování mé bakalářské práce.

Děkuji svým rodičům a příteli za jejich podporu a trpělivost, kterou se mnou měli během studia.

Abstrakt

V této bakalářské práci jsem se pokusila seznámit Vás se základními pojmy, které se vyskytují v oblasti spolehlivosti a zejména ve spolehlivosti software. Byl vysvětlen také vývoj samotné definice spolehlivosti.

Dalším důležitým tématem, kterým jsem se v této práci zabývala, byly základní metody analýzy spolehlivosti, kde jsem se věnovala jednotlivým metodám, například analýze způsobů a důsledků poruch (FMEA), analýze způsobů, důsledků a kritičnosti poruch, analýze stromu poruchových stavů (FTA), analýze blokovým schématem bezporuchovosti (RBD), Markovově analýze (MA) a dalším. Také jsem uvedla praktické příklady použití některých metod analýzy spolehlivosti.

Poté jsem se zabývala životním cyklem software. Uvedla jsem jeho etapy a základní požadavky na software. Dále jsem charakterizovala základní modely životního cyklu software. Za nejdůležitější považuji vodopádový životní cyklus, V - životní cyklus, rychlý vývoj aplikace a objektově orientovaný životní cyklus. V závěrečné části této práce jsem popsala dva typy modelů spolehlivosti software, a to model statický a dynamický.

Abstract

In this bachelor´s work I have attempted to acquaint you with the basic conceptions of reliability especially of software reliability. I have clarified the development of the reliability definition.

Another important theme I have dealt with in this work, was the basic methods of reliability analysis, where I have applied to the several methods. For example Fault Mode and Effects Analysis (FMEA), Fault Mode, Effects and Criticality Analysis (FMECA), Fault Tree Analysis (FTA), Reliability Block Diagram (RBD), Markov Analysis (MA), and other. Furthermore I have mentioned some examples of several methods.

Then I have dealt with software life cycle. I have mentioned software´s phases and requirements for software. Furthermore I have described the basic models of software life cycle. I rate the Waterfall Life-Cycle, the V Life Cycle, Rapid

Applications Development and Object Oriented Life Cycle as the most important models.

In the final part of this work I have described two types of reliability models. It is static and dynamic model.

Obsah

Seznam použitých symbolů a zkratek

ČSN	Česká technická norma
ETA	Event Tree Analysis - Analýza stromu událostí
FMEA	Fault Mode and Effects Analysis - Analýza způsobů a důsledků poruch
FMECA	Fault Mode, Effects and Criticality Analysis - Analýza způsobů, důsledků a kritičnosti poruch
FTA	Fault Tree Analysis - Analýza stromem poruchových stavů
HAZOP	Hazard and Operability Study - Studie nebezpečí a provozuschopnosti
HRA	Human Reliability Assessment - Posuzování spolehlivosti člověka
HW	Hardware
LBD	Logic Block Diagram - Logický blokový diagram
MA	Markov Analysis - Markovova analýza
MDT	Mean Down Time - Střední doba nepoužitelného stavu
MRT	Mean Repair Time - Střední doba opravy
MTBF	Mean Time Between Failure - Střední doba provozu mezi poruchami
MTTF	Mean Time to Failure - Střední doba do následujícího výpadku
MTTR	Mean Time to Repair - Střední doba do obnovy
MUT	Mean Up Time - Střední doba použitelného stavu
PC	Path County - Předpověď bezporuchovosti výpočtem z dílů
PHA	Preliminary Hazard Analysis - Předběžná analýza nebezpečí
RBD	Reliability Block Diagram - Analýza blokového diagramu bezporuchovosti
RPN	Risk Priority Number - Rizikové číslo
SW	Software

Úvod

K rozvoji spolehlivosti došlo v šedesátých letech dvacátého století. Spolehlivost je jedním z nejdůležitějších znaků jakosti produktu. Spolehlivost produktu je závislá na mnoha faktorech, a to na konstrukčním návrhu, způsobu provedení, a na správném zapojení všech prvků. K vytvoření spolehlivého systému, by se neměla zanedbat žádná etapa vývoje produktu. Samotnou spolehlivost lze hodnotit kvalitativně i kvantitativně pomocí ukazatelů spolehlivosti. [2]

Spolehlivost software (SW) je definována jako pravděpodobnost počítačového programu, že bude v provozuschopném stavu po určitou dobu [6]. O spolehlivosti software se začalo diskutovat v sedmdesátých letech dvacátého století, kdy se objevily první zmínky o jakosti SW a byl také kladen větší důraz na počáteční etapy životního cyklu SW. O spolehlivosti software se začalo diskutovat až v devadesátých letech dvacátého století, kdy došlo k velkému nárůstu využívání informačních technologií a tím pádem i software. V současnosti se využívá velké množství výpočetní techniky, a proto je důležité, aby byly spolehlivé.

Analýzy spolehlivosti slouží ke zjištění a prokázání základních ukazatelů jsko jsou ukazatelé bezporuchovosti, pohotovosti, udržovatelnosti, zajištěnosti údržby a bezpečnosti systému. K provedení systematické analýzy spolehlivosti je nutné používat jednotné postupy a správnou analytickou metodu spolehlivosti. Není vhodné používat pouze jednu metodu analýzy u složitějšího systému, žádná není natolik obecná, aby se mohla použít na jakýkoli systém. Existuje velké množství metod analýzy spolehlivosti, nejčastěji používanými bývají například analýza způsobů a důsledků poruch, analýza způsobů, důsledků a kritičnosti poruch, metoda blokových diagramů bezporuchovosti, metoda stromu poruchových stavů, ad. Podrobnější popis metod se nachází v normách ČSN IEC 60300. [16]

Výpočetní technika je v dnešní době velmi důležitá, a proto i vzrůstají požadavky na programové vybavení. Modely životního cyklu software znázorňují základní představu o tvorbě softwaru. Těchto modelů rozlišujeme větší množství, jako například vodopádový životní cyklus, V - životní cyklus, iterační životní cyklus, objektově orientovaný životní cyklus, atd. [4]

Doposud bylo vyvinuto velké množství modelů spolehlivosti software. Avšak nadále je nutné vyvíjet více praktických a realistických modelů k odhadnutí

spolehlivosti software. Testování software je účinný a nezbytný způsob, jak odstranit chyby v softwarovém produktu. Spolehlivost software lze posuzovat pomocí dvou typů modelů, a to modelu statického a dynamického. [9]

1 Terminologie používaná ve spolehlivosti

Spolehlivost procházela určitým vývojem, než se dostala do všech odvětví. Poprvé byla aplikována ve složitých zařízeních ve vojenství. V tomto odvětví došlo k rozvoji spolehlivosti již v padesátých letech dvacátého století a až poté se spolehlivost dostala i do jiných odvětví.

Počátek šedesátých let dvacátého století lze považovat za období, kdy docházelo k velkému rozvoji spolehlivosti. Dostala se do popředí kvůli rozvoji technických odvětví, především průmyslové výroby. Požadavky na spolehlivost rostou s ohledem k rozvoji procesů souvisejících se zajištěním vlastností objektu. Logicky by se předpokládalo, že s nárůstem počtu objektů by docházelo k většímu počtu poruch, ale toto se nelze zcela jednoznačně prokázat, protože kvalita objektů se obecně zvyšuje. V rámci některých analýz systémů je mnohem důležitější zabývat se analýzou důsledků možných poruch než faktem, že k poruchám může dojít. Tyto poruchy bylo nutné omezit, protože jejich následkem byly ekonomické ztráty, ale také ohrožení společnosti. Ohrožení společnosti je myšleno jak přímé (přímé dopady na zdraví a životy společnosti), tak nepřímé ohrožení (nefunkčnost systémů zásobování obyvatelstva). [1]

1.1 Definice spolehlivosti

Samotná definice spolehlivosti se postupem času měnila. Její první formulace zněla [1]: „Spolehlivost je pravděpodobnost, s jakou bude objekt schopen plnit bez poruchy požadované funkce po stanovenou dobu a v daných provozních podmínkách.“ Tato definice však nevystihovala spolehlivost v celé šíři. Anglické slovo *reliability* označovalo spolehlivost.

Druhá definice spolehlivosti zněla [1]: „Spolehlivost je obecná schopnost výrobku plnit požadované funkce po stanovenou dobu a v daných podmínkách, která se vyjadřuje dílčími vlastnostmi, jako jsou bezporuchovost, životnost, opravitelnost, pohotovost apod.“ Toto vymezení spolehlivosti se ještě tolik nepodobá současnému, jsou zde stále určité rozdíly. Anglické slovo *reliability* mělo v té době dva významy, jedním byla stále spolehlivost a druhým bezporuchovost.

Třetí definice zní následovně: „Souhrnný termín používaný pro popis pohotovosti a činitelů, které ji ovlivňují: bezporuchovost, udržovatelnost a zajištěnost údržby (používá se pouze pro obecný nekvantitativní popis).“ V období, kdy byla platná tato definice, se ještě používal termín spolehlivost údržby, který vyjadřoval schopnost poskytnutí služby na požádání uživatele a její zabezpečení po požadovanou dobu ve specifikovaných tolerancích a jiných daných podmínkách. Nově používaný anglický výraz pro spolehlivost zní *dependability*. [8]

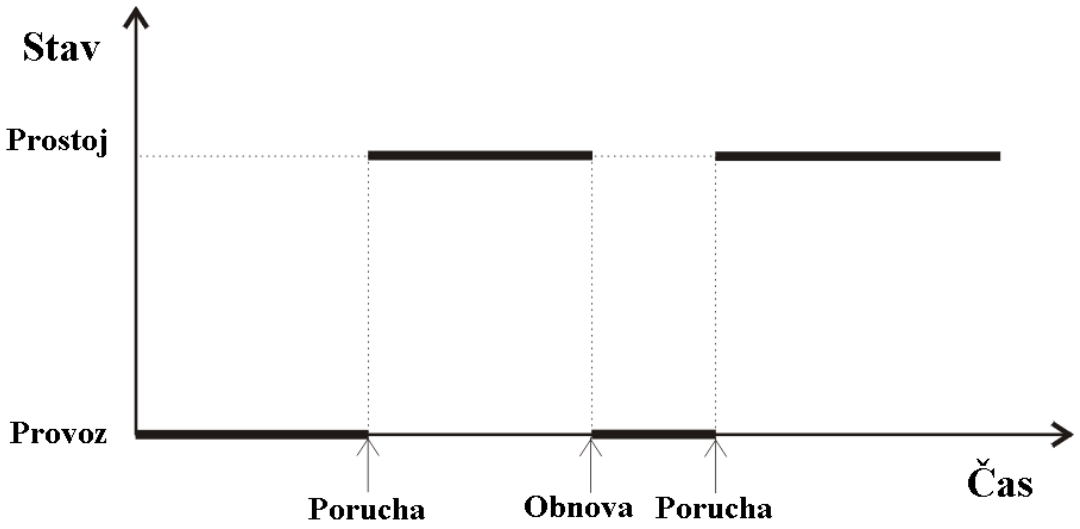
Čtvrté vymezení spolehlivosti je prozatím ve fázi návrhu a připomínkuje se. Předpokládá se, že čtvrté znění definice spolehlivosti bude následující. Spolehlivost je schopnost kdykoli plnit požadované funkce. Toto vymezení bude patrně zahrnovat předpoklad, že plnění požadovaných funkcí významně ovlivňují i vnější činitelé. Kvantifikovat a vyjádřit spolehlivost číselným ukazatelem nelze, tento pojem se používá pouze pro obecný popis. Jednotlivým činitelům je možné přiřadit numerickou hodnotu, je možné je kvantifikovat. Mezi tyto činitele patří pohotovost, bezporuchovost, udržovatelnost a zajištěnost údržby. Spolehlivost může být používána s různými přívlastky, mezi nejčastěji používané patří spolehlivost inherentní, provozní a odhadovaná (neboli predikovaná). Za inherentní (vloženou) spolehlivost se považuje ta, která je do objektu vložena v průběhu návrhu a výroby objektu. Provozní spolehlivostí se rozumí spolehlivost, u které se uvažují provozní vlivy a další podmínky. Predikovanou (odhadovanou) spolehlivost můžeme definovat jako výsledek výpočtů, analýz a prognóz spolehlivosti projektovaného objektu. [8]

1.2 Základní pojmy ve spolehlivosti

K porozumění spolehlivosti je důležité znát některé základní pojmy, které se v této oblasti vyskytují. Přehled těchto základních pojmů následuje níže: [1], [8]

- *Sledované objekty* - Objekt dle normy je definován následovně: jakákoliv část, součást, zařízení, část systému nebo celý systém, s kterým je možné se individuálně zabývat. Sledované objekty mohou být tedy takové, u kterých se posuzuje spolehlivost (hodnocení se vždy vztahuje k plnění požadované funkce). Těmito objekty mohou být výrobky nebo služby. Objekt se může skládat z hardwaru, softwaru nebo z obojího současně (objektem může být i člověk, u kterého je jeho spolehlivost rovněž posuzována, ale specifickými metodami).

- *Vlastnosti objektu* - Spolehlivost je jednou z vlastností objektu, která se definuje nejčastěji. K popisu spolehlivosti slouží pohotovost, bezporuchovost, udržovatelnost a zajištěnost údržby. Životnost, efektivnost a způsobilost jsou doplňkové vlastnosti, které se používají pouze v některých rozšířených úvahách o spolehlivosti. Efektivnost je schopnost objektu vyhovět požadavkům na služby s danými kvantitativními charakteristikami (tato schopnost však závisí na kombinaci hledisek způsobilosti a pohotovosti objektu). [8] Způsobilost je schopnost objektu plnit požadavky na služby s danými kvantitativními charakteristikami při daných vnitřních podmínkách.
- *Jevy a stavy* - objekt se může z pohledu spolehlivosti nacházet v několika různých stavech. Mohou to být například tyto stavy: provoz, prostoj, provozuneschopný stav, nepoužitelný stav, atd. Ke stavům objektu se vztahují časové údaje, kterými mohou být: doba provozu, doba prostoje, požadované funkce, atd. V praxi se můžeme setkat s více případy, kdy objekt slouží k popisu vzájemných souvislostí mezi stavem a dobou, na kterou se váže. Charakteristickým příkladem je, že objekt je popisován diskrétními stavy a spojitým časem. Systémy obnovované opravou se mohou nacházet minimálně ve dvou stavech, a to ve stavu provozu (stav, kdy objekt plní požadovanou funkci) nebo prostoje (stav, kdy objekt neplní požadovanou funkci). Přechod mezi těmito stavy u systémů obnovovaných opravou se nazývá procesem prosté obnovy. Proces prosté obnovy může vypadat následovně, viz. Obrázek číslo 1. [8]



Obrázek 1: Prostý proces obnovy

- *Náhodné proměnné* - ve spolehlivosti souvisí s náhodným výskytem sledovaných jevů a kvantitativního popisu. Náhodné proměnné jsou spojovány se sledovanou veličinou, obvykle časem - dobou. Těmito proměnnými může být doba provozu, doba údržby, doba opravy, apod.
- *Ukazatele spolehlivosti* - Pomocí ukazatelů můžeme spolehlivost posuzovat kvantitativně. Definice jednotlivých ukazatelů nalezneme v normě ČSN IEC 600 50(191). [8] Za nejdůležitější mohou být považovány následující: funkce okamžité pohotovosti, funkce okamžité nepohotovosti, součinitel střední pohotovosti, součinitel střední nepohotovosti, pravděpodobnost bezporuchového provozu, intenzita poruch, střední intenzita poruch, parametr proudu poruch, střední doba provozu mezi poruchami, střední doba do poruchy, střední doba do obnovy, atd. Jednotlivé ukazatele spolehlivosti mohou být získány zpracováním náhodného pokusu, náhodného jevu, pravděpodobnosti nebo náhodné proměnné. Grafickým znázorněním může být například distribuční funkce nebo graf hustoty pravděpodobnosti.

1.3 Činitele spolehlivosti

Spolehlivost lze komplexně posuzovat pomocí jednotlivých dílčích činitelů spolehlivosti. Jednotlivé činitele spolehlivosti (bezporuchovost, udržovatelnost a

zajištěnost údržby) jsou popsány níže, mohou mít různou podobu, a to kvalitativní nebo kvantitativní.

1.3.1 Bezporuchovost

Bezporuchovost (anglicky *reliability*) je chápána jako schopnost objektu plnit požadovanou funkci ve stanovených podmínkách a během specifikovaného časového intervalu. Bezporuchovost systému popisují ukazatelé bezporuchovosti. Jsou jimi například pravděpodobnost bezporuchového provozu, intenzita poruch, střední intenzita poruch, střední doba provozu mezi poruchami (MTBF - Mean Time Between Failure), ad. [2]

1.3.2 Udržovatelnost

Udržovatelnost je definována následovně [1], [8]: „Schopnost objektu v daných podmínkách používání setrvat ve stavu nebo vrátit se do stavu, v němž může plnit požadovanou funkci, jestliže se údržba provádí v daných podmínkách a používají se stanovené postupy a prostředky.“ Ukazateli udržovatelnosti je například okamžitá intenzita opravy, střední intenzita opravy, střední doba opravy (anglická zkratka MRT), střední doba do obnovy (MTTR - *Mean Time to Repair*), střední pracnost údržby. [2]

1.3.3 Zajištěnost údržby

Vymezení pojmu zajištěnost údržby je dáno následovně: „Schopnost organizace poskytující údržbářské služby zajišťovat podle požadavků v daných podmínkách prostředky potřebné pro údržbu podle dané koncepce údržby.“ [8]

Existují tři typy údržby, a to údržba (obecně), údržba po poruše a preventivní údržba. Vymezení těchto typů údržby jsou níže. [5]

Definici údržby je formulována následovně: „Kombinace všech technických a administrativních činností, včetně činnosti dozoru, zaměřených na udržení objektu ve stavu nebo jeho navrácení do stavu, v němž může plnit požadovanou funkci.“ [8]

Pojem údržby po poruše zní takto: „Údržba prováděná po zjištění poruchového stavu a zaměřená na uvedení objektu do stavu, v němž může plnit požadovanou funkci.“ [8]

Definice preventivní údržby je formulována následovně: „Údržba prováděná v předem určených intervalech nebo podle předepsaných kritérií a zaměřená na snížení pravděpodobnosti poruchy nebo degradace fungování objektu.“ [8]

1.4 Požadavky na spolehlivost

Pokud chceme zajistit, aby systém, který vytváříme, byl spolehlivý, musíme pečlivě vytvořit návrh a stanovit kvalitativní a kvantitativní požadavky na spolehlivost. Jestliže budeme požadavky na spolehlivost zanedbávat již od návrhu systému, není možné vytvořit projekt, který by byl spolehlivý. Tyto požadavky si může určovat výrobce, odběratel nebo případně může výrobce a odběratel na nich spolupracovat, aby byl projekt co nejvíce kvalitní a spolehlivý. [1]

Požadavky na spolehlivost se zadávají pro jednotlivé prvky systému, tyto jednotlivé požadavky musí samozřejmě splňovat i výsledný systém. V normě ČSN IEC 600 50 se nacházejí podrobnější informace o specifikaci požadavků na spolehlivost. Dále v této normě jsou rady, které pomáhají zákazníkům i dodavatelům, tak aby došlo ke kompromisu mezi oběma stranami. [8]

K tomu, aby byly dosaženy požadavky spolehlivosti, musí forma požadavků obsahovat následující fakta: [1]

- *Zvolené ukazatele spolehlivosti* - jde o výčet zvolených ukazatelů spolehlivosti (např. funkce okamžité pohotovosti, funkce okamžité nepohotovosti, součinitel střední pohotovosti,…)
- *Číselné hodnoty těchto ukazatelů* - číselné ohodnocení jednotlivých ukazatelů spolehlivosti.
- *Správné provozní podmínky objektu* - jakých hodnot by měly ukazatele spolehlivosti nabývat.
- *Časové období života objektu*
- *Kritéria poruchy a mezních stavů* - vymezení rysů jednotlivých poruch a mezních stavů
- *Zásady pro ověřování spolehlivosti*

V současnosti se požadavky na spolehlivost dají rozdělit do následujících samostatných skupin: [1], [2], [8]

- *Požadavky na pohotovost* - Pokud kvantitativní požadavky na spolehlivost nejsou dostatečně specifikovány, musí se definovat i kvalitativní. U složitých technických systémů se mohou vyskytovat poruchové stavy, které neovlivňují schopnost plnit požadovanou funkci. Mezi ukazatele pohotovosti můžeme zařadit funkci okamžité pohotovosti, součinitel střední pohotovosti, střední doba použitelného stavu, ad.
- *Požadavky na bezporuchovost* - mohou být kvantitativní nebo kvalitativní. Kvalitativní požadavky se mohou určovat pomocí kritérií návrhu výrobku nebo jako činnosti pro zlepšení bezporuchovosti. Kvantitativní požadavky lze vyjádřit pomocí číselných hodnot ukazatelů bezporuchovosti, kterými mohou být například pravděpodobnost bezporuchového provozu, okamžitá intenzita poruch, střední doba do poruchy, ad.
- *Požadavky na udržovatelnost* - zahrnují požadavky na testovatelnost, kvantitativní a kvalitativní ukazatele udržovatelnosti. Kvantitativními ukazateli mohou být následující: střední doba preventivní údržby, střední doba opravy, střední doba do obnovy, ale také mohou klást požadavky na pracnost údržby, na počet pracovníků údržby, atd. Kvalitativní požadavky se většinou týkají konstrukčního provedení objektu nebo způsobu, jakým se samotná údržba projektu provádí.
- *Požadavky na zajištěnost údržby* - ukazatele, které používá udržovatelnost, se mohou využívat i pro zajištěnost údržby, kvantitativní jsou ale doplněny např. o střední logistické zpoždění, střední administrativní zpoždění, aj.

1.5 Spolehlivost software

Spolehlivost software (SW) je definována jako pravděpodobnost počítačového programu, že bude v provozuschopném stavu po určitou dobu. [6]

O spolehlivosti software se začalo hovořit v sedmdesátých letech dvacátého století, kdy se objevily první zmínky o jakosti SW a byl také kladen větší důraz na počáteční etapy životního cyklu SW. O spolehlivosti jako takové se začalo diskutovat až v devadesátých letech dvacátého století, kdy došlo k velkému nárůstu využívání

výpočetní techniky a tím pádem i software. V současnosti se využívá velké množství informačních technologií, a proto je důležité, aby byly spolehlivé.

Dříve bylo hardwarové vybavení výpočetní techniky mnohem důležitější než softwarové. Nyní tomu je přesně naopak. Jestliže srovnáme vývoj hardware (HW) a software zjistíme, že na špičkovém produktu se podílí přibližně 20 návrhářů z oblasti HW a 100 inženýrů z oblasti SW.

Pro měření spolehlivosti se využívá střední doba mezi výpadky systému (MTBF - Mean Time between Failures), která je dána součtem střední doby do obnovy (MTTR - Mean Time to Recovery) a střední dobou do následujícího výpadku (MTTF - Mean Time to Failure). Pokud se do softwarového produktu často přidávají nové funkce a pokud je v tomto produktu mnoho chyb, které musejí být odstraňovány, tak mnohdy dochází ke snižování spolehlivosti software.

1.5.1 Pohotovost

U jakýchkoli projektů (jak hardwarových tak softwarových) je důležité, aby systém byl schopný vykonávat určenou funkci kdykoli je potřeba a za stanovených podmínek. Tato vlastnost může být pro zákazníka jedna z nejdůležitějších. Pohotovost může být predikována na základě dlouhodobého pozorování.

Pohotovost, i přes mírně rozdílné pojetí, jak u software, tak u hardware, je možné rozdělit na pohotovost inherentní (teoretická) a operační (provozní), pokud chceme její požadavky kvantifikovat. Inherentní součinitel pohotovosti je vyjádřen pomocí střední doby mezi poruchami (MTBF) a střední doby do obnovy (MTTR). Pohotovost operační se vztahuje ke skutečnému provoznímu prostředí. Součinitel pohotovosti je kombinací střední doby použitelného stavu (MUT - Mean Up Time) a střední doby nepoužitelného stavu (MDT - Mean Down Time).

1.5.2 Údržba software

Na vývoj software se nyní vydává mnohem více nákladů než na vývoj hardware. Pro srovnání, v roce 1955 se za vývoj SW utratilo 20% z celkové ceny výpočetní techniky, v roce 1985 to bylo již 90% z celkové ceny. Dříve byla údržba software pouze malou komponentou životního cyklu software, nyní je tomu naopak, organizace za údržbu software utrácejí velké sumy. [3]

Důležitá fakta o údržbě SW: [3]

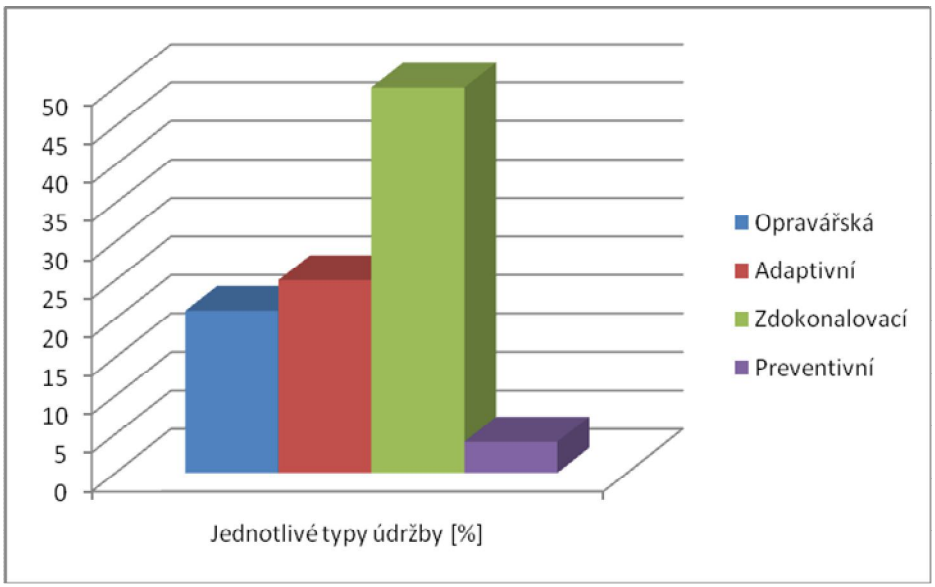
- Studie organizace Hewlett-Packard odhalila, že přibližně 60 - 80% jejich zaměstnanců, kteří pracují v oblasti výzkumu a vývoje, je zapojeno do údržby již existujícího software.
- Přes 40% aktivit spojených s údržbou je kvůli modifikacím a nastavením, které požadují zákazníci.
- Typické rozpětí životního cyklu software obvykle bývá takové, že jeden až tři roky stráví ve vývoji a obvykle deset až patnáct let v užívání (včetně údržby).
- Za každý dolar, který se utratí ve vývoji software, se musí počítat s dalším, který udrží SW v provozu po celý jeho životní cyklus. Navíc může být další dolar utracen za změny žádané od zákazníků.
- Údržba software stojí přibližně 70% ceny SW.
- Údržbou existujícího SW můžeme strávit až 60% všeho úsilí spojeného s vývojem.

Softwarovou údržbu můžeme rozdělit do čtyř skupin, které představují druhy údržbových prací: [3]

1. *Opravárenská (opravářská) údržba* - úkolem tohoto typu údržby je odstraňování nalezených chyb.
2. *Adaptivní údržba* - jejím cílem je přizpůsobení softwaru změnám prostředí (bez změny funkce systému). Za změnu prostředí může být považováno zapojení nového hardware nebo použití aktualizované verze operačního systému.
3. *Zdokonalovací údržba* - zabývá se řešením požadavků uživatele (většinou jde o funkční zlepšení systému), snaží se o zdokonalení uživatelského rozhrání a zvýšení výkonnosti systému.
4. *Preventivní údržba* - má za úkol zvýšit udržovatelnost systému (tj. aktualizování dokumentace, doplnění komentářů,…)

Rozložení jednotlivých druhů údržby je na obrázku číslo 2. Procentuálně jsou jednotlivé typy údržby zastoupeny takto: opravářská údržba - 21%, adaptivní údržba - 25%, zdokonalovací údržba - 50% a preventivní údržba - 4%. Toto rozložení se postupem let mění, ale jen nepatrně. Avšak podíl údržby vzhledem k ostatním činnostem roste. Je to důsledkem toho, že pojem údržba zahrnuje i vývoj. Pokud

chceme, aby prováděná údržba byla smysluplná a neměnila jakost softwarového produktu, musíme zajistit její správný průběh.



Obrázek 2: Podíl jednotlivých údržbových prací

1.6 Dílčí závěr

V této části práce byl vysvětlen samotný pojem a význam spolehlivosti. Také byly zmíněny jednotlivé činitele spolehlivosti, a to bezporuchovost, udržovatelnost a zajištěnost údržby, tyto činitele mohou mít podobu kvalitativní i kvantitativní. Bylo poukázáno na význam požadavků na spolehlivost. Poté následoval popis spolehlivosti software, kde byly uvedeny základní údaje a samozřejmě i její definice. Údržba software je považována za důležitou komponentou životního cyklu software.

V této části byla přiblížena problematika spolehlivosti. Pro další vzdělávání v této oblasti by bylo vhodné použít další odbornou literaturu.

2 Základní metody analýzy spolehlivosti

Analýzy spolehlivosti slouží ke zjištění a prokázání ukazatelů bezporuchovosti, pohotovosti, udržovatelnosti a bezpečnosti systému. K provedení systematické analýzy spolehlivosti je nutné používat jednotné postupy a správnou analytickou metodu spolehlivosti. Není vhodné používat pouze jednu metodu analýzy u složitějšího systému, žádná není natolik obecná, aby se mohla použít na jakýkoli systém.

Principielní provedení jednotlivé konkrétní analýzy spolehlivosti z hlediska přístupu je možné realizovat jedním ze dvou základních způsobů. Prvním z nich je přístup induktivní. Podstatou induktivního přístupu je fakt, že analýzu provádíme od základních problémů k obecným. Postupuje se od analýzy jednoho prvku k analýze celého systému. Například v analýze způsobů a důsledků poruch se využívá induktivního přístupu. Druhým možným způsobem je deduktivní přístup. Provádí analýzu od globálních problémů k základním. Uskutečňuje analýzu opačným postupem než induktivní přístup.

2.1 Obecné fáze postupu analýzy spolehlivosti

Analýzu spolehlivosti můžeme obecně rozdělit na čtyři hlavní fáze, které jsou popsány a definovány níže. [2]

- *Funkční a technická analýza* - v tomto stádiu analýzy dochází ke shromažďování dat, zjišťování vlastností, cílů, funkčních a technických charakteristik. Také se zde provádí první funkční analýza, která slouží k získání informací týkajících se provozních podmínek a funkcí. Funkční a technická analýza by měla usnadňovat kvalitativní a kvantitativní analýzu tím, že budou shromážděna všechna data a potřebné údaje.
- *Kvalitativní analýza* - v této fázi se provede kvalitativní analýza, která by nás měla informovat o všech poruchách, příčinách těchto poruch a o důsledcích, které přinášejí.
- *Kvantitativní analýza* - zde se provádí odhad číselné hodnoty ukazatelů spolehlivosti. Je nutné, aby analytik správně posoudil dobu provozu, způsob ověřování funkce, aby také ověřil zásady průběhu preventivní a nápravné údržby a také musí korektně určit rozsah a rychlost změny provozních

podmínek. Tato analýza se u jednodušších systémů realizuje ručně a složitějších je nutné použít speciální program, který je určený k tomuto výpočtu.

- *Syntéza výsledků analýzy* - jak již je z názvu patrné, jde o závěrečnou část analýzy. Analytik provede syntézu informací ze závěrů z kvalitativní a kvantitativní analýzy. Určí poruchy, na kterých je spolehlivost systému nejvíce závislá. Z toho plynou určitá technická a technologická opatření, která vedou ke zvýšení úrovně spolehlivosti a bezpečnosti. Pak ještě je nutné provést změny, jež vedou k úpravám údržbových operací, ale také ke zmenšení rizika vlivu lidského faktoru.

Etapy životního cyklu hardware a doporučené aplikace použitelných metod analýzy spolehlivosti v jednotlivých etapách může vypadat následujícím způsobem:

- Koncepce a stanovení požadavků - v této etapě životního cyklu hardware se stanoví funkce a požadovaná úroveň spolehlivosti systému. Provádí se zde kvalitativní metody - předběžná analýza nebezpečí (PHA) a konstrukční FMEA/FMECA.
- Návrh a vývoj - zde se provádí konstrukční zpracování systému a mohou se realizovat zkoušky spolehlivosti. Obvykle jsou prováděny kvantitativní metody analýzy, které logicky navazují na analýzy předešlé etapy. Mezi charakteristické metody prováděné v této etapě patří například metoda blokového diagramu bezporuchovosti (RBD), analýza stromem poruchových stavů (FTA), Markovova analýza (MA) a procesní FMEA/FMECA.
- Výroba - během této etapy životního cyklu HW se neprovádí žádná analýza spolehlivosti.
- Instalace - stejně jako u etapy výroby ani zde neprobíhá žádná analýza spolehlivosti.
- Provoz a údržba - po dobu záruky systému (nebo po celý život systému) shromažďuje výrobce data z provozu, které je možné statisticky zpracovat pomocí klasických metod statistické analýzy a tím získat cenné informace o spolehlivosti objektu v provozu.
- Vypořádání - systém dosáhne mezního stavu.

2.2 Přehled vybraných analýz spolehlivosti

Existuje velké množství metod analýzy spolehlivosti, které lze s ohledem na vyjádření v kapitole 2.1 s úspěchem aplikovat. Nejčastěji používanými analýzami bývají například analýza způsobů a důsledků poruch, resp. analýza způsobů, důsledků a kritičnosti poruch, metoda analýzy blokových diagramů bezporuchovosti, metoda analýzy stromu poruchových stavů, atd. Podrobnější popis metod se nachází v normách ČSN IEC. Níže jsou jednotlivé metody stručně vysvětleny.

Jednotlivé metody mohou mít kvalitativní, kvantitativní nebo semikvantitativní povahu. Kvalitativní metody jsou takové, jejichž výsledkem jsou nečíselné charakteristiky. Kvantitativní metoda je taková, jež je číselně ohodnocena - zjišťujeme pravděpodobnost. Za semikvantitativní metodu považujeme každou metodu, která je jak číselně, tak i slovně ohodnocena.

2.2.1 Analýza způsobů a důsledků a poruch (FMEA)

Analýza způsobů a důsledků poruch, zkratka FMEA značí počáteční písmena z anglického Fault Mode and Effects Analysis. Poprvé byla použita v šedesátých letech dvacátého století na projektu APOLLO agentury NASA. Jde o kvalitativní metodu s induktivním přístupem. [1]

Metoda se používá k nalezení samotných poruch a následně ke zjištění jejich příčin a důsledků. Základní způsoby užití metody FMEA je následující:

- *FMEA konstrukční* - vzhledem ke konstrukčnímu návrhu se uvažují poruchové stavy, které mohou nastat. Poté se zkoumají jejich příčiny a následky, změnou konstrukčního návrhu dochází ke snaze potlačit většinu poruchových stavů.
- *FMEA procesní (výrobní)* - Navazuje na FMEA konstrukční a měla by odstranit poruchové stavy, které jsou zapříčiněné nedostatky v návrhu výrobního nebo montážního procesu.
- *FMEA systémová* - je komplexnějším pojetím FMEA procesní.

Pro správné provedení metody FMEA bychom se měli řídit určitým postupem, který může vypadat následovně. FMEA může být řešena jak jednotlivcem, tak pracovním týmem (tento tým může mít například pět až sedm členů, ve kterém by měli být odborníci z různých oborů). Poté tým shromažďuje základní podklady a potřebné

informace. V dalším kroku provádíme samotnou metodu. Nutné je předem stanovit hloubku analýzy. U všech prvků stanovujeme důsledky, způsoby a příčiny poruch. Kvalitativně posoudíme významnost poruch. Následně vyhodnotíme závěry, které by měly vést k nápravným opatřením (odstranění příčin poruch, snížení pravděpodobnosti vzniku poruchy). Zhodnotíme, zda požadavky, které jsme stanovili, byly splněny. Pokud jsou výsledky analýzy neuspokojivé, je možné provést analýzu ještě jednou v jiné hloubce. Tato metoda se využívá během prvních dvou etap životního cyklu hardware, a to při koncepci a stanovování požadavků, návrhu a vývoji.

Podrobnější informace o této metodě nalezneme v normě ČSN IEC 60 812. [17]

2.2.2 Analýza způsobů, důsledků a kritičnosti poruch (FMECA)

Analýza způsobů, důsledků a kritičnosti poruch - FMECA (Fault Mode, Effects and Criticality Analysis) rozšiřuje metodu FMEA pouze o hodnocení kritičnosti důsledků. Kritičnost se zabývá poruchami a řetězením následných poruch, které mohou způsobit velké škody. FMECA se za určitých podmínek může řadit mezi semikvantitativní metody (tzn., že metoda se dá řešit kvantitativně i kvalitativně). [1]

Postup metody FMECA je podobný jako u metody FMEA. Nejdříve zjistíme informace (popis, příčiny, důsledky poruch) o všech poruchových stavech, které mohou nastat. Dalším krokem k úspěšnému provedení analýzy je ohodnocení současného stavu tzv. rizikovým číslem (RPN), které se vypočte součinem bodového ohodnocení pravděpodobnosti výskytu poruchy, bodového ohodnocení závažnosti následku a bodového ohodnocení detekce příčiny/následku poruchy. Poté by měla být prodiskutována a realizována nápravná opatření (změnou konstrukčního plánu nebo výrobního postupu). Po provedení změn se opět ohodnotí současný stav pomocí rizikového čísla pro zjištění, zda nápravná opatření vedla k vyřešení problému a byla efektivní.

Stejně jako metoda FMEA se i FMECA využívá během prvních dvou etap životního cyklu hardware, a to při koncepci a stanovování požadavků, návrhu a vývoji.

Stejně jako analýza způsobů a důsledků poruch, tak i FMECA je popsána v normě ČSN IEC 60 812. [17]

2.2.3 Analýza stromem poruchových stavů (FTA)

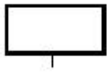
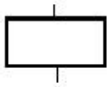
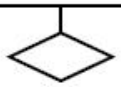
Analýza stromem poruchových stavů - FTA (Fault Tree Analysis) je na rozdíl od metod FMEA a FMECA deduktivní analýza. FTA se řadí mezi kvalitativní metody analýzy spolehlivosti. Metoda FTA může identifikovat kombinaci základních selhání zařízení a lidských chyb, které mohou vést k poruchovému stavu. Je vhodná pro analýzu rozsáhlých systémů. Určuje druhy poruchových stavů, jejich příčiny, vzájemné vazby mezi poruchovými stavy a snaží se nalézt vrcholovou událost. [1]

Analýza se realizuje pomocí stromu poruchových stavů (je tedy znázorněna graficky), postupuje se až do nejnižší úrovně systému a zde se určí příčina nebo příčiny poruch. Při tvorbě stromu poruchových stavů se užívají schematické značky, na obrázku číslo 3 jsou některé ze základních znázorněny, ve skutečnosti jich je více.

Tuto metodu je možné aplikovat během druhé etapy životního cyklu hardware, a to při návrhu a vývoji. Podrobnější popis analýzy stromu poruchových stavů se nachází v normě ČSN EN 61 025. [18]

Postup metody FTA může vypadat následovně:

- *Definování vrcholové události (Top Event)* - událost vyplývající z účinku všech vnějších a vnitřních podmínek. Většinou jde o negativní událost, snažíme se jí zabránit.
- *Určení příčin poruchových stavů*
- *Zobrazení výsledků pomocí stromu poruchových stavů* - grafické zobrazení logických vazeb mezi vrcholovou událostí a dalšími událostmi popisujícími její vznik.
- *Možnost provedení kvantitativní analýzy* - pokud chceme získat kvantitativní výsledky analýzy, tak uvedeme u každé události ve stromu poruchových stavů pravděpodobnost výskytu.

Doporučená značka	Alternativní značka	Název a popis
		Blok s názvem nebo popisem vrcholové události (TOP jevu).
		Blok s názvem nebo popisem události (jevu), případně s uvedením pravděpodobnosti výskytu (pokud se to požaduje).
		Základní (primární) událost – událost, která se dále nedělí.
		Nerozvíjená událost – událost, která není dále rozvíjená (zpravidla proto, že se to nepovažuje za nutné)

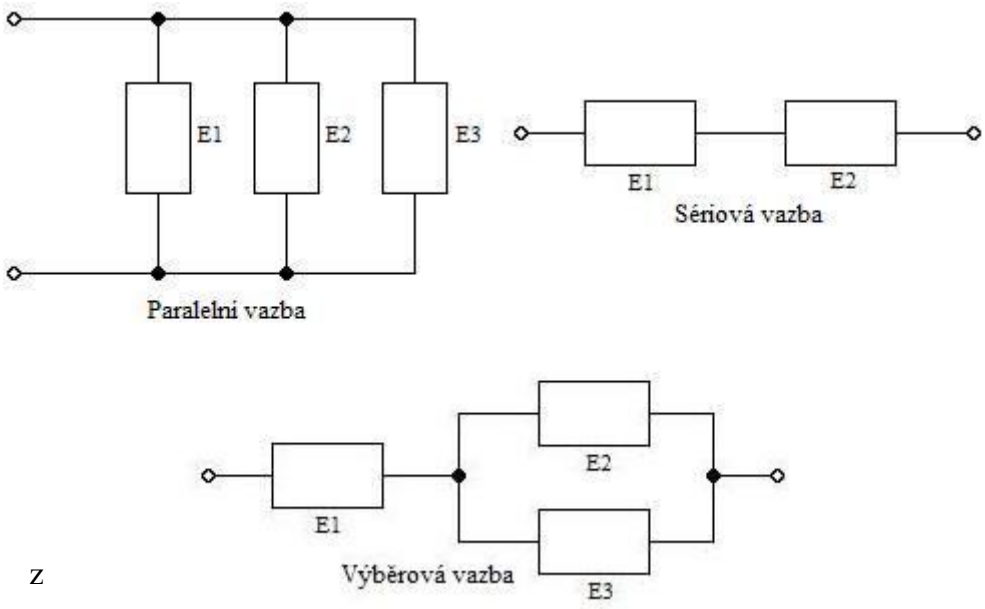
Obrázek 3: Základní schematické značky stromu poruchových stavů

Metodu FTA lze řešit pomocí software, programy nám pomohou s výpočty ukazatelů spolehlivosti. Mezi vhodné softwarové produkty řadíme: Item Software, Risk Spectrum, Relex Software a další. [7]

2.2.4 Analýza blokového diagramu bezporuchovosti (RBD)

Analýza blokového diagramu bezporuchovosti - RBD (Reliability Block Diagram) je deduktivní metodou analýzy bezporuchovosti. Základem je logický blokový diagram (LBD - Logic Block Diagram). Ke grafickému zobrazení struktury systému se využívá blokového schématu. Metoda se využívá ke zjišťování hodnot ukazatelů bezporuchovosti. Zobrazuje logické spojení funkčních vazeb prvků, jež jsou uspořádány ve struktuře systému a představují při bezporuchovém fungování správnou funkci systému. [1]

Vazba mezi prvky systému může být sériová, paralelní nebo výběrová. Na obrázku číslo 4 jsou zobrazeny jednotlivé vazby.



z

Obrázek 4: Vazby mezi prvky systému

Existují dvě základní metody řešení systémů, a to metoda dekompozice systému a inspekční metoda. Metoda dekompozice systému lze použít jen u takových systémů, u kterých jsou jednotlivé prvky nezávislé. Nejdříve celý systém rozdělíme na části skládající se ze sériových a paralelních soustav a určíme u nich pravděpodobnost. Po spojení jednotlivých soustav získáme výslednou pravděpodobnost. U inspekční metody stav systému vyjádříme jako logickou kombinaci jevů vyjadřujících stavy jednotlivých prvků a dále vyšetříme, s jakou pravděpodobností tato kombinace jevů může nastat.

Tuto metodu je možné aplikovat během druhé etapy životního cyklu hardware, a to při návrhu a vývoji.

O analýze blokového diagramu bezporuchovosti se více dozvíme v normě ČSN IEC 61 078. [19]

2.2.5 Markovova analýza (MA)

Základem Markovovy analýzy - MA (Markov Analysis) je teorie Markovových řetězců a procesů. Hledáme základní ukazatele (např. pravděpodobnost) toho, že prvky systému, resp. systém jako celek, jsou v určitém okamžiku v některém z možných stavů. MA využívá různé přístupy, převážně využívá induktivní přístup. Je možné ji používat u systémů, u kterých je nutné brát v úvahu pořadí vzniku poruchových stavů.

Využívá se za účelem modelování systému, pomocí kterého by mělo být zjištěno jeho chování. Také by se pomocí této analýzy mohly odhadnout ukazatele bezporuchovosti, pohotovosti udržovatelnosti a bezpečnosti.

Tuto metodu je možné aplikovat během druhé etapy životního cyklu hardware, a to při návrhu a vývoji.

Detailnější charakteristika Markovovy analýzy se nachází v normě ČSN IEC 61 165. [20]

2.2.6 Předpověď bezporuchovosti výpočtem z dílů (PC)

Předpověď bezporuchovosti výpočtem z dílů - PC (Path County) je induktivní metodou. PC je součástí Military Handbook 217F a doplňkovou metodou Path County je metoda Part Stress - Předpověď bezporuchovosti výpočtem z namáhání. PC se provádí již během konstrukčního návrhu. Systém se musí rozdělit na primární komponenty, u těchto prvků se zjistí intenzita pomocí knihy Military Handbook 217F, kde jsou uvedeny vzorce, podle kterých se tato intenzita počítá. Výsledkem analýzy je odhad intenzity poruch systému s předpokladem, že poruchu způsobí jakýkoli prvek systému. Metoda PC se využívá pro odhadu intenzity poruch elektronických zařízení a systémů. Analýzu lze provádět pomocí softwarové aplikace.

Využití této metody vzhledem k informačním technologiím nalezneme u hardware. [1][2][5]

2.2.7 Příklady některých dalších analýz spolehlivosti

Metod analýzy spolehlivosti je mnoho. Dalšími důležitými jsou například následující:

- *Studie nebezpečí a provozuschopnosti* - HAZOP je anglickou zkratkou pro *Hazard and Operability Study*. Metoda hledá problémy, které se týkají nebezpečí a provozuschopnosti. [2]
- *Analýza stromu událostí* - je značena zkratkou ETA z anglického *Event Tree Analysis*. ETA se znázorňuje graficky. K provedení analýzy se využívá induktivní přístup. Tato metoda je založena na zjednodušených principech analýzy stromu poruchových stavů. [2]
- *Předběžná analýza nebezpečí* - metoda se značí zkratkou PHA (*Preliminary Hazard Analysis*). Slouží k identifikaci situací, jež mohou způsobit poškození systému. [2]
- *Posuzování spolehlivosti člověka* - anglicky *Human Reliability Assessment* - HRA. Jedná se o kvalitativní metodu. A jejím cílem je posouzení vlivu chybných činností (lidských chyb), kterých se člověk dopouští, na spolehlivost systému. [2]

2.3 Příklady aplikace metod analýzy spolehlivosti

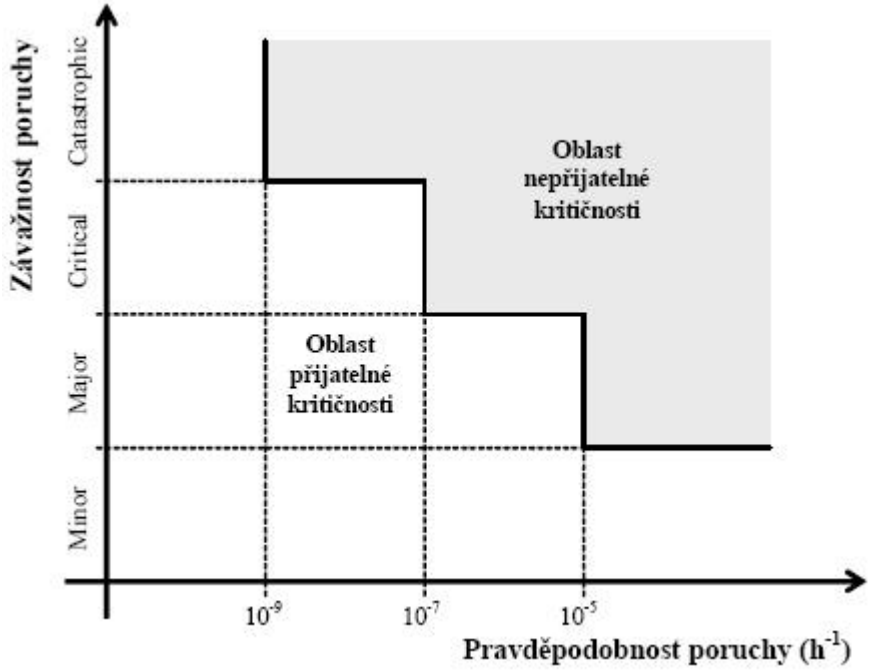
Z důvodu vyjádření návaznosti výše uvedených metod analýz spolehlivosti na téma práce jsou níže uvedeny některé praktické příklady jejich aplikace:

2.3.1 Použití FMEA/FMECA

FMEA / FMECA - rozsah užití této metody je široký, příkladem může být využití této metody například na antivirovém softwaru. Jednotlivými funkcemi - body metody FMEA/FMECA může být: špatná elektroinstalace, prasknutí žárovky, tepelná deformace, zatékání - problém s vlhkostí,… U každé funkce se určí popis poruchového stavu, projev, možný důsledek, příčina.

Rizikové číslo se vypočte součinem bodového ohodnocení pravděpodobnosti výskytu poruchy (*A*), závažnosti následku (*B*) a schopnost detekce příčiny/následku poruchy (*C*). *A* můžeme ohodnotit jedním až deseti body, kde jeden bod znamená, že poruchový stav není pravděpodobný a deset bodů má význam vysoké pravděpodobnosti výskytu. *B* lze ohodnotit také jedním až deseti body. Jeden bod určuje zanedbatelné následky a deset bodů značí velmi vysoké následky. *C* také může nabývat jednoho až deseti bodů. Jeden bod získá ta porucha, u které je velmi vysoká pravděpodobnost

detekce poruchy již v době návrhu, a deset bodů bude mít porucha s velmi vysokou pravděpodobností odhalení až u koncového zákazníka. Rizikové číslo může nabývat hodnot jedna až tisíc. Kritičnost poruchy se vyhodnocuje pomocí grafu kritičnosti poruch, tento graf kritičnosti může vypadat následovně viz. obrázek číslo 5. [5]



Obrázek 5: Graf kritičnosti poruch

Na následujícím obrázku (číslo 6) je zobrazena tabulka, která znázorňuje praktické využití metody FMEA/FMECA pro návrh výrobku. Je zde provedena analýza antivirového software. Jde o příklad triviální, ale dostatečně názorný. Rizikové číslo získáme součinem hodnot význam, výskyt, odhalitelnost. Příslušné hodnoty nalezneme v tabulce. Po provedení nápravných opatření se provede analýza opětovně - rizikové číslo by se mělo zmenšit. [14]

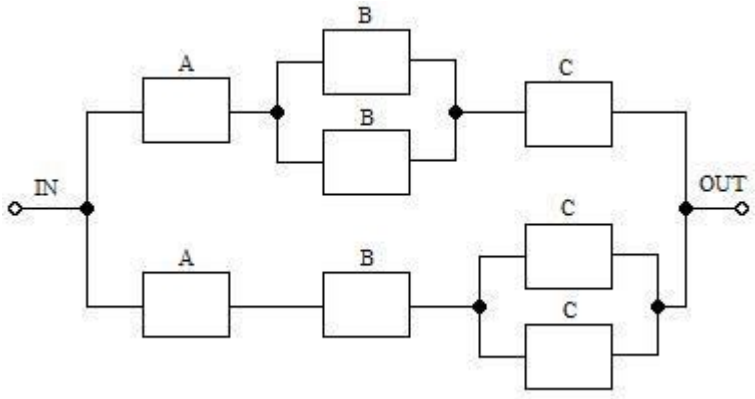
Prvek	Možná vada	Možné následky vady	Význam	Kritičnost	Možné příčiny	Výskyt	Stávající způsoby posuzování návrhu	Odhalitelnost	Rizikové číslo	Doporučená opatření	Provedená opatření	Význam	Výskyt	Odhalitelnost	Rizikové číslo
Antivirový software	Nezná všechny viry	Poškození mnoha souborů Postoupení viru dalším prostřednictvím internetu	10		Programátor nepočítal s novými typy virů	10	Žádné	10	1000	Aktualizace databáze virů	Databáze se aktualizuje každý den	10	2	2	40
	SW neumožní otevření nakaženého souboru	Nemožnost odstranění viru Ztráta cenných dat	8		Nevhodné naprogramování	4	Žádné	6	192	Možnost otevření souboru s výstrahou	Je možné otevřít nakažený soubor	2	2	4	16

Obrázek 6: Příklad metody FMEA/FMECA

2.3.2 Příklad RBD

Příklad využití analýzy blokovým schématem bezporuchovosti je znázorněn na následujícím příkladu.

Příklad: Vypočtěte, s jakou pravděpodobností v čase 10 000 hodin nedojde k poruše zařízení, které se skládá výhradně ze software. Software, který slouží k malování, má tři prvky, které jsou na sobě závislé. Intenzita prvku A je rovna $0{,}5 \cdot 10^{-5} h^{-1}$, prvku $1 \cdot 10^{-5} h^{-1}$ a prvku C $1{,}5 \cdot 10^{-5} h^{-1}$. Na obrázku číslo 7 je znázorněna souvislost mezi jednotlivými prvky.



Obrázek 7: Příklad zapojení zařízení

Nejdříve vypočteme pravděpodobnosti prvků A, B a C:

$$R_A(10000) = e^{-\lambda \cdot t} = e^{-0,5 \cdot 10^{-5} * 10000} = e^{-0,05} = 0,9512$$
$$R_B(10000) = e^{-0,1} = 0,9048$$
$$R_C(10000) = e^{-0,15} = 0,8607$$
$$R_{BB}(10000) = 1 - [(1 - R_B) \cdot (1 - R_B)] = 0,9909$$
$$R_{CC}(10000) = 1 - [(1 - R_C) \cdot (1 - R_C)] = 0,9806$$
$$R_I(10000) = R_A \cdot R_{BB} \cdot R_C = 0,8112$$
$$R_{II}(10000) = R_A \cdot R_B \cdot R_{CC} = 0,8439$$
$$R_{CELK} = 1 - [(1 - R_I) \cdot (1 - R_{II})] = 0,9705$$

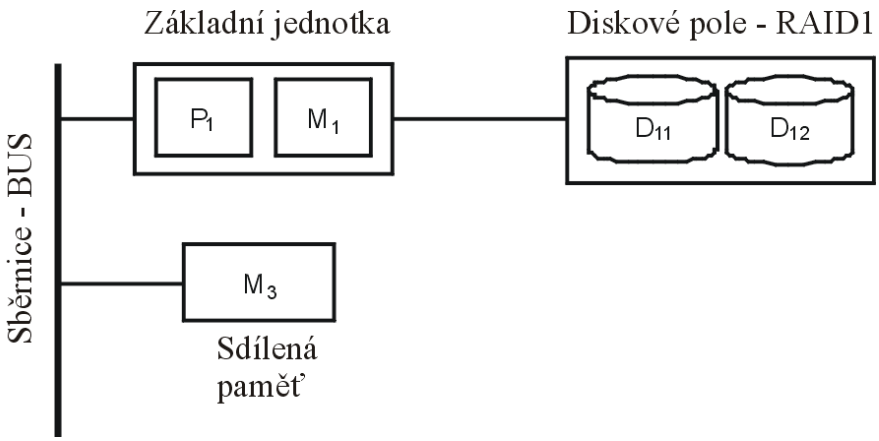
Poznámka: R_{BB} a R_{CC} jsou pravděpodobnosti dvou paralelně zapojených prvků.

S pravděpodobností 97,05% nedojde do času 10000 hodin k poruše softwarového zařízení.

2.3.3 Příklad použití MA

ITEM Toolkit Software představuje výpočetní modul markovské analýzy. Tento software je určený pro analýzy spolehlivosti systémů, jejichž chování lze popsat markovskými procesy. Spolehlivost analyzovaného systému je definována konečnou množinou stavů a množinou přechodů mezi zvolenými dvojicemi stavů. Ke konstrukci markovského modelu se používá grafický editor, který umožňuje zadat přechodový diagram včetně jeho spolehlivostních parametrů.

Praktickým příkladem využití Markovovy analýzy může být analýza jednoprocesorového systému. Tento příklad je prezentovaný na HW, avšak i v hardwarových systémech jsou softwarové součásti. Schematické znázornění tohoto systému je na obrázku číslo 8.



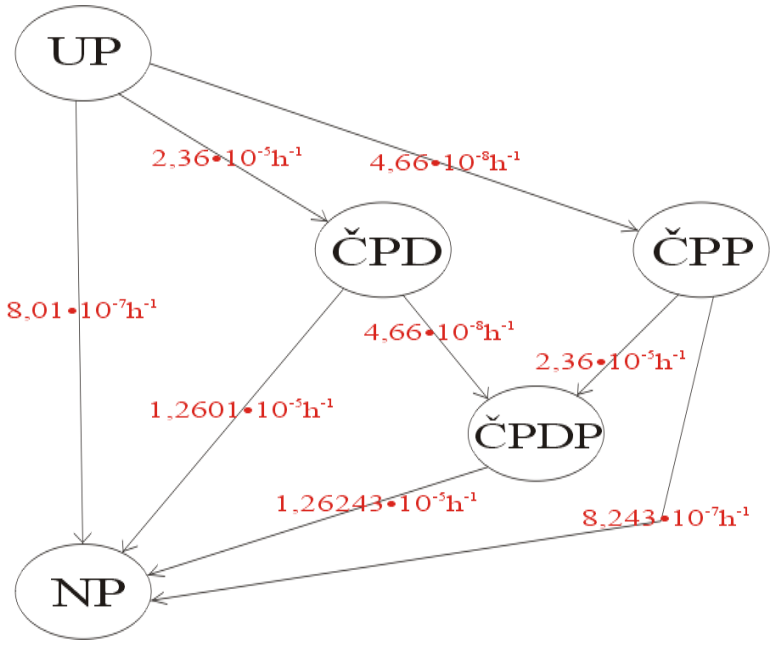
Obrázek 8: Jednoprocesorový systém

Systém je v provozuschopném stavu, jestliže je současně v provozuschopném stavu procesor (P_1), sběrnice (BUS), alespoň jedna z pamětí M_1, M_3 (výběrový systém 1 z 2) a alespoň jeden z disků D_{11}, D_{12} (výběrový systém 1 z 2).

Parametry jednotlivých prvků jsou následující:

- Pevný disk: intenzita poruch je $4{,}35 \cdot 10^{-5} h^{-1}$, intenzita oprav $4{,}17 \cdot 10^{-2} h^{-1}$.
- Paměť: intenzita poruch je $2{,}91 \cdot 10^{-8} h^{-1}$, intenzita oprav $1{,}04 \cdot 10^{-2} h^{-1}$.
- Procesor: intenzita poruch je $1{,}35 \cdot 10^{-5} h^{-1}$, intenzita oprav $1{,}04 \cdot 10^{-2} h^{-1}$.
- Sběrnice: intenzita poruch je $1{,}45 \cdot 10^{-7} h^{-1}$, intenzita oprav $5{,}95 \cdot 10^{-3} h^{-1}$.

Spolehlivost uvažovaného systému je popsána následujícím přechodovým diagramem - obrázek číslo 9. [15]



Obrázek 9: Přechodový diagram

Popis přechodového diagramu:

- UP - stav úplné provozuschopnosti (všechny prvky systému jsou v provozuschopném stavu)
- ČPD - stav částečné provozuschopnosti (jeden disk v poruchovém stavu)
- ČPP - stav částečné provozuschopnosti (jedna paměť v poruchovém stavu)
- ČPDP - stav částečné provozuschopnosti (jeden disk a jedna paměť v poruchovém stavu)
- NP - neprovozuschopný stav

Orientované hrany vyjadřují možnosti přechodů mezi stavy a jejich ohodnocení reprezentuje příslušnou intenzitu přechodu.

2.4 Dílčí závěr

Existuje velké množství známých a frekventovaně využívaných metod analýz spolehlivosti, které se ale převážně zabývají stanovením spolehlivosti hardware. Metody, které jsou považovány za jedny z nejdůležitějších, jsou popsány výše, přičemž naproti tomu metody vhodné pro posuzování spolehlivosti software jsou dále uvedeny v kapitole 4. Dále jsou uvedeny u některých metod praktické příklady využití. Nejde o podrobné seznámení s každou metodou, jsou zde formulována pouze základní fakta.

Uvedla jsem obecné fáze postupu analýzy spolehlivosti, kde jsem jednotlivé fáze popsala. Poté jsem pomocí etap životního cyklu hardware poukázala na provázanost jednotlivých fází spolehlivosti a etap životního cyklu hardware. Metody PHA a konstrukční FMEA/FMECA se využijí v počáteční fázi životního cyklu - v etapě koncepce a stanovení požadavků. Metody FTA, RBD, MA a procesní FMEA/FMECA se aplikují během etapy návrhu a vývoje. V etapě provozu a údržby se využívají klasické metody statistické analýzy.

Využití zmíněných metod je široké, můžeme je použít v mnoha odvětvích, například pro posouzení spolehlivosti elektronických součástek, softwarového či hardwarového vybavení výpočetní techniky, atd.

3 Životní cyklus software

Výpočetní technika je v dnešní době velmi důležitá, a proto i vzrůstají požadavky na programové vybavení. Níže jsou popsány etapy a následně modely životního cyklu softwaru. Životním cyklem softwaru se zabývá obor softwarové inženýrství. Softwarové inženýrství zahrnuje činnosti jako inženýrství, informatiku a management. Jeho cílem je návrh, tvorba a údržba programového vybavení výpočetní techniky. [4]

3.1 Požadavky na software

Požadavkem bychom měli chápat schopnost nebo vlastnost, kterou má software mít, aby uživatel mohl vyřešit problém - dosáhnout cíl. Definované požadavky by měly být úplné a bezesporné.

Požadavky je možné rozdělit na funkční a mimofunkční. Funkční požadavky jsou takové, které popisují funkce nebo služby, jež jsou od softwaru očekávány. Mimofunkční požadavky se netýkají funkce software, ale vztahují se k vlastnostem jako je například spolehlivost. V některých případech jsou mimofunkční požadavky důležitější než funkční (například pokud bude řídicí systém automobilu nespolehlivý, tak je takové zařízení nepoužitelné).

Další možné dělení požadavků na software je dle specifikace, a to specifikace uživatelské a systémové. Pod uživatelskou specifikací požadavků je chápán popis funkčních a mimofunkčních požadavků zákazníka, tyto požadavky by měly být srozumitelné pro všechny uživatele. Systémová specifikace požadavků je podrobnější specifikace uživatelských požadavků pro vývojáře.

3.2 Etapy životního cyklu softwaru

Životní cyklus software můžeme rozdělit do jednotlivých etap - fází. K tomu, aby vznikl kvalitní software, by nejdříve měl projít několika etapami: [4]

1. *Specifikace a analýza požadavků* - v této etapě by se měly specifikovat požadavky a následně by mělo dojít k jejich analýze. Nemělo by docházet k realizaci požadavků.

2. *Návrh* - cílem druhé etapy je vytvoření systémového návrhu a návrhu jednotek, kde by se měly podrobně specifikovat softwarové součásti, struktura dat, ad. Měly by se zde odhadnout náklady a doba trvání projektu.
3. *Programování* - ve třetí etapě dochází k samotnému programování softwarových součástí.
4. *Testování* - zde dochází k testování celého systému, pokud se vyskytne chyba, tak by mělo dojít k její nápravě. Poté se systém testuje uživatelem, pokud je systém schválen, tak dochází k instalaci systému a následně ke školení samotných uživatelů.
5. *Provoz a údržba* - další etapou životního cyklu softwaru je jeho provoz a údržba. Měly by zde být řešeny problémy, které souvisejí s používáním softwaru. V této etapě stráví software nejvíce času (přibližně 60%) z celého životního cyklu.
6. *Mezní stav* - poslední etapou životního cyklu software je dosažení mezního stavu, kdy je software vyřazen z provozu. Důvodem vyřazení SW z provozu může být fakt, že je neaktuální či zastaralý. Například iterační životní cyklus předpokládá, že software bude vyřazen z provozu.

Software by se měl dělit do pěti primárních procesů. Primárními procesy životního cyklu jsou: proces akvizice, dodání, vývoje, provozování a údržby. Dále ještě existují podpůrné procesy životního cyklu a organizační procesy životního cyklu. [21]

3.3 Modely životního cyklu softwaru

Modely životního cyklu software znázorňují základní představu o tvorbě softwaru. Mohou být označovány jako konceptuální modely tvorby software. Těchto modelů rozlišujeme větší množství, jako například vodopádový životní cyklus, V - životní cyklus, iterační životní cyklus, objektově orientovaný životní cyklus, ad. Základní charakteristiku některých životních cyklů softwaru nalezneme níže. [4]

Většinou jsou upřednostňovány ty modely, které mohou provádět zpětné kroky (jako například model kontinuálního testování). Možnost provedení zpětného kroku může přinést několik výhod a samozřejmě i nějaké nevýhody. Za výhodu můžeme považovat menší organizační úsilí, pokud se však vyskytne problém se špatnou kvalitou produktu, s překročením nákladů nebo s nedodržením termínů, tak náklady vynaložené

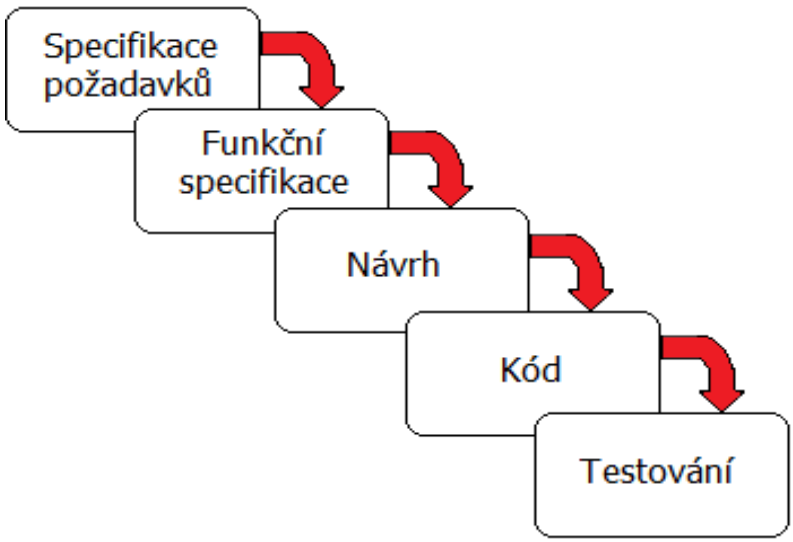
na jejich nápravu mohou být velmi finančně náročné. Velkou nevýhodou může být nemožnost jakostního řízení celého projektu, příčinou této nevýhody může být neplánování procesu tvorby software (z hlediska časových termínů, ale i z hlediska nákladů). [4]

3.3.1 Vodopádový životní cyklus

Anglický výraz The Waterfall Life-Cycle označuje vodopádový životní cyklus. Tento model je považován za jeden nejstarších a nejčastěji užívaných. Vodopádový životní cyklus je velmi jednoduchý pro porozumění, ale i pro užívání.

Vodopádový model by měl projít pěti fázemi v následujícím pořadí: specifikace požadavků, funkční specifikace, návrh, kód a testování. Princip vodopádového modelu je takový, že k další fázi modelu je možné přistoupit pouze, když je aktuální fáze kompletně dokončena. Na konci každé fáze by mělo dojít k hodnocení s cílem zjistit, zda je projekt na správné cestě k jeho úspěšnému dokončení nebo, jestliže je nutné, aby došlo k nápravným opatřením. Na obrázku číslo 10 je vodopádový model znázorněn.

Výhodou modelu je jednoduché a snadné použití. Jednotlivé fáze a procesy jsou zpracovány a ukončovány vždy jedna po druhé. Vodopádový životní cyklus lze použít i pro menší projekty. Tento model není vhodné aplikovat, pokud existuje riziko, že budou požadovány nějaké změny.

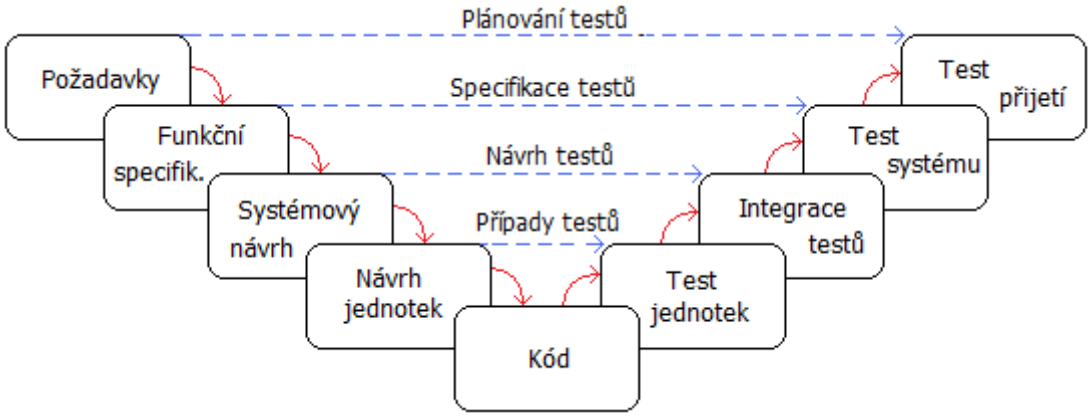


Obrázek 10: Vodopádový životní cyklus

3.3.2 V – životní cyklus

V - životní cyklus (The V Life Cycle) je variantou vodopádového modelu. Proto i u tohoto modelu platí, že k další fázi je možné přistoupit až po dokončení stávající fáze. U tohoto modelu by mělo po skončení jednotlivých fází docházet k testování z důvodu dosažení vysoké jakosti software. Na obrázku číslo 11 je zobrazen průběh V - životního cyklu.

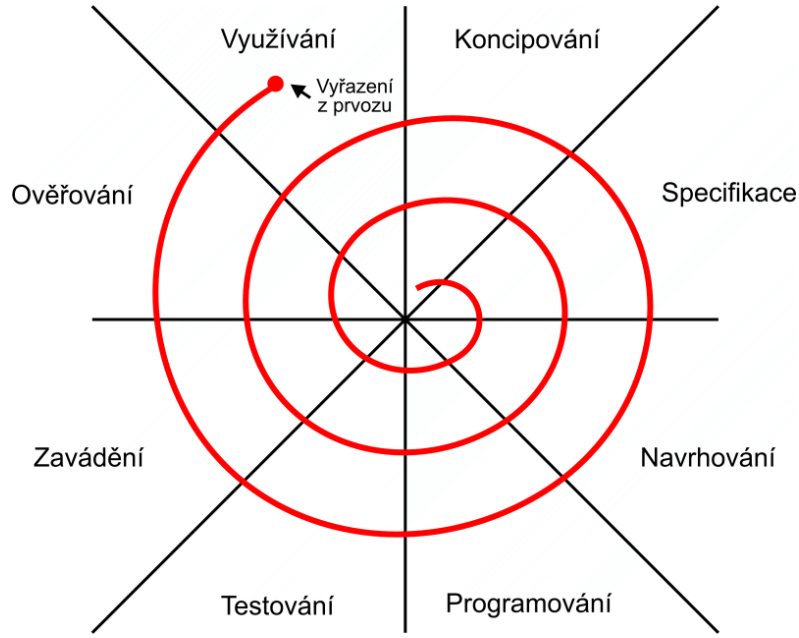
Výhodou modelu je jednoduché použití, větší šance úspěšného provedení projektu než u vodopádového modelu kvůli rozvoji plánu testování systému již na počátku životního cyklu software. Nevýhodou může být nemožnost provádění zpětných kroků. Dalším negativním faktem je, že není vyroben žádný prototyp softwaru během počátečních fází.



Obrázek 11: V - životní cyklus

3.3.3 Iterační životní cyklus

Anglicky Iteration Life Cycle. Vývoj u iteračního modelu prochází několika opakujícími se cykly. Během jednotlivých cyklů se zvyšuje kvalitativní úroveň. Je zde zdůrazněn fakt, že vývoj softwaru probíhá postupně. Software několikrát prochází následujícími fázemi: koncipování, specifikace, navrhování, programování, testování, zavádění, ověřování, využívání. Vývoj produktu je pozastaven okamžikem, kdy se vyřadí z používání. Na obrázku číslo 12 je graficky znázorněn průběh iteračního životního cyklu.



Obrázek 12: Iterační životní cyklus

3.3.4 Objektově orientovaný životní cyklus

Objektově orientovaný životní cyklus (anglicky An Object Oriented Life Cycle) souvisí s objektově orientovaným přístupem tvorby software. Tento model by mohl být v budoucnosti jedním z nejvíce užívaných modelů. Softwarová aplikace je modelována jako množina spolupracujících objektů.

Model postupně prochází fázemi, jako je specifikace požadavků, návrh systému, vytvoření knihovny tříd, návrh objektů, kód objektů, test objektů, test funkcí a poslední fází je finální přijetí softwaru jako celku.

3.3.5 Rychlý vývoj aplikace

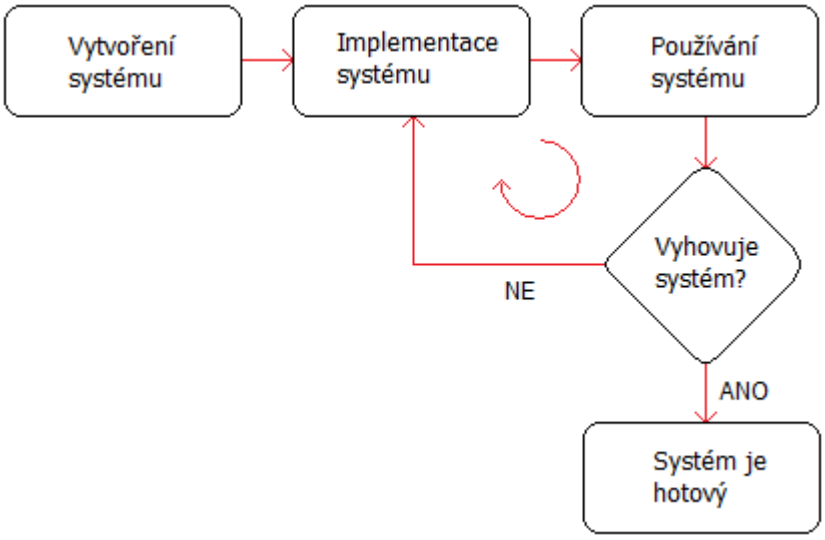
Rychlý vývoj aplikace (anglicky Rapid Applications Development) patří mezi novější - modernější trendy vývoje softwaru. Tento model se zaměřuje na rychlost, snaží se zkrátit dobu vývoje softwaru. Nejdříve se vytvoří základní jádro aplikace, které se postupně rozšiřuje. Mezitím však dochází k využívání již vytvořených softwarových funkcí.

3.3.6 Průzkumný vývoj

Průzkumný vývoj (anglicky Exploratory Development) je založen na následujícím postupu: nejdříve vytvoříme počáteční verzi softwarové aplikace, ke které uživatel sestaví komentáře, poté jsou vytvářeny další verze, které jsou upravovány až do odpovídající úrovně. Postupně se aplikaci přidávají nové vlastnosti. Důležité je spolupracovat s uživatelem, aby vznikla požadovaná softwarová aplikace.

Jednou z výhod průzkumného vývoje může být přizpůsobitelnost dodatečným požadavkům uživatele. Jednou z nevýhod může být těžké finanční, časové a personální plánování.

Na obrázku číslo 13 nalezneme graficky znázorněný postup průzkumného vývoje.

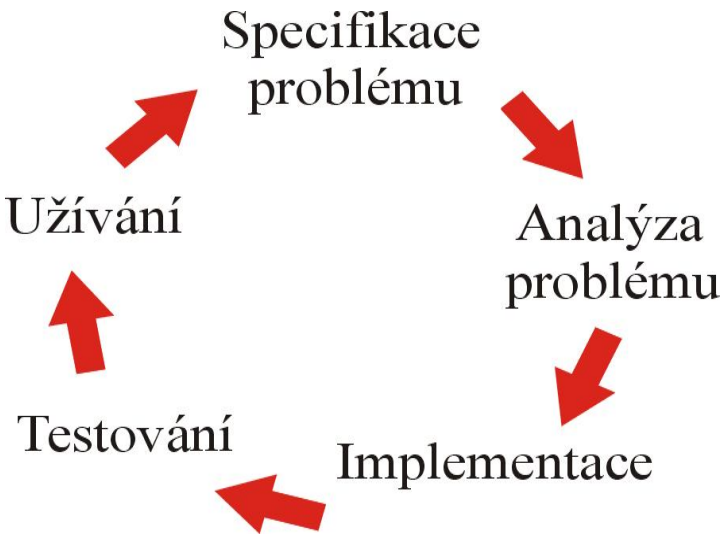


Obrázek 13: Průzkumný vývoj

3.3.7 Kruhový životní cyklus

Popisuje cyklickou tvorbu jednotlivých verzí softwarových produktů. Model prochází pěti fázemi, a to specifikace problému, analýza problému, implementace, testování a užívání. Model je graficky znázorněn na obrázku číslo 14.

Software se bude používat jen se základními funkcemi, postupně (již během užívání software) se budou doplňovat další funkce.



Obrázek 14: Kruhový životní cyklus

3.3.8 Model kontinuálního testování

Řeší návratové kroky na základě vyhodnocení výsledků po ukončení každé fáze. Autorem kontinuálního modelu testování (anglicky *Continuous Testing Model*) je japonský profesor Fuji.

3.3.9 Model prototyp

U modelu prototyp (anglicky *Throw-away Prototyping*) se nejdříve vytvoří prototyp, který se po ověření funkcí přestane používat. Cílem tohoto modelu by mělo být lepší porozumění zákazníkových požadavků. Výhodou je testování softwaru budoucími uživateli, kteří specifikují své požadavky. Nevýhodou tohoto modelu mohou být vyšší náklady než u jiných modelů.

3.4 Dílčí závěr

Celá kapitola prezentuje několik možností přístupu a implementace v rámci životního cyklu software. Na základě průzkumu dostupných zdrojů je uvedeno několik nejvýznamnějších zástupců struktury životního cyklu software. U nově vytvářeného

software je důležité určit požadavky, které budeme očekávat. Je na každém vývojáři respektive uživateli, jaký model životního cyklu si vybere.

Modelů životního cyklů software existuje velké množství, není možné zde charakterizovat všechny. Jsou zde charakterizovány takové, které považuji za důležité. Mezi nejčastěji využívané patří vodopádový model, objektově orientovaný životní cyklus, atd.

4 Možnosti posuzování spolehlivosti software

Doposud bylo vyvinuto velké množství modelů spolehlivosti software, a to užíváním nehomogenních Poissonových procesů, Markovových procesů, Bayesovy statistiky, atd. S uvedenými metodami se můžeme setkat například v publikacích od autorů Musa, Xie [13], Goel, Pham [22], atd. Testování software je účinný a nezbytný způsob, jak odstranit chyby v softwarovém produktu. [9]

Jednotlivé modely spolehlivosti software by měly odpovědět na následující otázky: [9]

- Jaká by byla intenzita poruch, kdyby byl software užíván nyní?
- Kolik poruch zůstává v software? Kolik jich je velmi vážných? Jaká je lokalizace poruchy?
- O kolik více testování je potřeba k dosažení cílů spolehlivosti software? Je softwarový produkt dostatečně spolehlivý na to, aby byl uvolněn k užívání?

Poptávka po komplexním softwarovém systému vzrostla rychleji než schopnost navrhování, implementace, testování a údržby. Spolehlivost SW systémů se stala hlavním zájmem pro naši moderní společnost. [11]

Spolehlivost software je měřena počtem provozních poruch nebo počtem poruch, které jsou zpozorovány ve vývoji spolu s různými vedlejšími informacemi. [10] Vedlejší informace zahrnují čas, ve kterém byla porucha zjištěna, část softwaru, ve kterém byla porucha rozpoznána, a také stav softwaru v době zpozorování poruchy nebo vedlejší informací může být povaha poruchy. [10]

Existuje velké množství významných rozdílů mezi softwarovými a hardwarovými systémy. Protože teorie spolehlivosti softwaru má velký vliv na hardware, je důležité znát tyto rozdíly, aby bylo možné vytvořit realistické modely pro software. Základní rozdíly mezi softwarem a hardwarem mohou být následující:

- Software zastarává v jiné intenzitě než hardware. Nelze říci, že rychleji ani pomaleji, ale prostě jinak. Velmi však záleží na konkrétních podmínkách. Obecně ale platí, že díky možnosti modulace a modifikace vzniká u SW v pozdějším věku méně poruch než u HW.
- Pokud je jednou vymazána chyba ze softwarového programu, tak již nikdy nezpůsobí tu samou poruchu. [13]

4.1 Modely spolehlivosti software

Možnosti pro modelování spolehlivosti software svým počtem pravděpodobně převyšují množství modelů spolehlivosti hardware (modely spolehlivosti hardware jsou jednoznačně vnímány jako předchůdci modelů spolehlivosti software). Samozřejmě existuje mnoho rozdílů mezi těmito dvěma modely. Poruchy software mohou být zapříčiněné chybnými požadavky, návrhem, zdrojovým kódem nebo neschopností spolupracovat s ostatními souvisejícími softwary. Spolehlivost software se většinou zvyšuje s dobou užívání, u hardware je tomu naopak. [10]

Existují dva typy modelů spolehlivosti software, a to statický a dynamický odhad spolehlivosti. Tyto modely spolehlivosti se většinou využívají v počátečních ale i pozdějších fázích vývoje SW. [10]

4.1.1 Statické modely

Jedním z cílů statických modelů spolehlivosti může být provedení odhadu spolehlivosti již v počáteční fázi vývoje SW. Tato činnost určuje budoucí spolehlivost software založenou na dostupných softwarových metrikách a mírách. Zejména když nejsou k dispozici poruchová data (například když je software ve fázi návrhu nebo kódování), tak metriky, které se získají z procesu vývoje software a z charakteristik výsledného produktu, mohou být využity k odhadu spolehlivosti softwaru během testování. [11]

Existuje velké množství modelů spolehlivosti, mezi statické modely spolehlivosti software náleží fázově založený model a model prediktivního vývoje životního cyklu. Níže nalezneme zformulovaná základní fakta o těchto modelech: [10], [11]

- *Fázově založený model* - Tento model byl vytvořen již v roce 1988 Gaffneym a Davisem. Tento model dělí vývoj software do jednotlivých fází, například do těchto: přehled požadavků, návrh, implementace, testování, integrace software, provoz, atd. Model používá statistické přehledy poruch, které se získají na počátku vývojové fáze (například během technického přezkoumání požadavků, návrhu, a předpovědi spolehlivosti během návrhu a provozu). Předpokládá, že odhady velikosti kódu jsou k dispozici již během počátečních fází vývoje. S fázově založeným modelem souvisí s Raleighovou hustotou

pravděpodobnosti funkce. Tento model je zřejmě motivovaný modelem, který se užívá ve spolehlivosti hardware.

- *Prediktivní vývoj životního cyklu* - O vytvoření prediktivního vývoje životního cyklu se zasloužili Dalal a Ho. V tomto modelu je vývoj životního cyklu rozdělen do stejných fází jako u fázově založeného modelu, avšak není zde stanovena pevně daná vazba mezi počtem poruch objevených během různých fází. Základním předpokladem pro tento model může být následující fakt. Hodnocení závad odlišných produktů ve stejné fázi životního cyklu jsou vzorky ze statistického souboru produktů, které pocházejí z vývojové instituce. Tzn., že produkty vyvíjené stejnou institucí ve stejném stádiu životního cyklu jsou více či méně stejné.

4.1.2 Dynamické modely

Dynamické modely spolehlivosti software mohou být nazývány modely růstu spolehlivosti pro testování a využití v provozu. Pro model růstu spolehlivosti software se používá anglický termín Software Reliability Growth Model se zkratkou SRG. Zahrnuje několik desítek modelů, například Musa lineární model, nebo Musa-Okumoto logaritmický model, Jelinski-Moranda model, Goel-Okumoto model, atd. Základem SRG je zkoumání výskytu poruch a snaží se předpovědět jejich budoucí chování. [10], [12].

Odhad spolehlivosti software je určen aktuální spolehlivostí software (spolehlivost software se získá využitím odvozených statistických metod) a poruchovými daty (poruchová data se získají během testování systému nebo během jeho provozu). Mnoho aktuálních modelů spolehlivosti software patří do této kategorie. [10]

Většina modelů růstu spolehlivosti je založena na společných základních předpokladech. Mohou to být následující předpoklady: [10]

- Systém zůstává během testování v podstatě nezměněný s výjimkou vymazání nalezených chyb. Některé modely počítají s možností, že některé chyby nemusejí být opraveny správně.
- Odstraňování chyb nemá vliv na možnost, že jiná chyba bude nalezena.
- Čas - doba je měřena takovým způsobem, že testovací výkon je konstantou.

- Všechny chyby jsou stejně důležité (protože se ve stejné míře podílejí na poruchovosti).
- Na začátku testování je určitý celkový počet chyb, který může být fixní nebo libovolný.

Pokud chceme použít modely růstu spolehlivosti, tak bychom měli být opatrní na následující fakta. Software by neměl pracovat v odlišném prostředí a odlišným způsobem než v jakém byl testován. Jestliže software bude do jisté míry pracovat rozdílně, poté předpověď spolehlivosti nebude přesná a může dojít i k dalším problémům.

Dynamické modely spolehlivosti se mohou používat k rozhodování, kdy ukončit testování. Testování spolehlivosti je nezbytný, ale nákladný proces, který utratí jednu třetinu až jednu polovinu ceny typického vývojového projektu. Testování rozsáhlých softwarových systémů stojí až tisíce dolarů denně. Přehnané testování může vést k produktu, který je předražený a pozdě uvedený na trh, zatímco fixace chyb je dražší u systému, který je v provozu, na rozdíl od systému, u kterého fixace chyby proběhla během testování. Z toho vyplývá, že otázka, jak moc testovat, je důležitá z ekonomického rázu.

4.2 Dílčí závěr

O trochu více pozornosti je v poslední době věnováno spolehlivosti v různých průmyslových odvětvích. Je to způsobeno tím, že o kvalitní software je stále větší zájem.

Základní rozdělení některých modelů spolehlivosti software může být rozdělení na statické a dynamické modely. Oba dva typy zahrnují velké množství modelů.

K odhadnutí spolehlivosti software se samozřejmě dají využívat i základní metody analýzy spolehlivosti, které jsem charakterizovala v kapitole 2. Mezi tyto metody patří například analýza způsobů a důsledků poruch, analýza způsobů, důsledků a kritičnosti poruch, Markovova analýza, atd.

Závěr

Velké množství informačních technologií je v současnosti využíváno k podpoře hardwarových systémů, které dříve fungovaly autonomně. Proto je důležité, aby kombinace nových technologií a jejich vlivů měly spolehlivý průběh a dopad. Dříve byl hardwarovému vybavení systémů přikládán větší význam z důvodu neexistence podpory elektronikou a informačními technologiemi. Aplikace výpočetní techniky však získává stále větší zázemí do mnoha, dříve čistě strojních, aplikací. Principy těchto systémů jsou vnímány jako velmi pozitivní a to z důvodů vyšších výkonů, řízení, bezpečnosti, optimalizace funkce, úspor, ekologie, a jiných. Nyní se tedy jeví, že software a jeho převaha v technickém světě bude pokračovat.

V současné době začínají být pouze hardwarové nebo softwarové systémy pomalu v menšině, stále častěji existují systémy smíšené (mechatronické), které mají dříve zmíněné dvě složky - softwarovou a hardwarovou. Z tohoto důvodu se pro analýzu spolehlivosti využívají metody určené pro software současně s metodami určenými pro analýzu spolehlivosti hardware.

V poslední době je o trochu více pozornosti věnováno spolehlivosti v různých průmyslových odvětvích. Je to způsobeno tím, že o kvalitní software je stále větší zájem.

Seznam použité literatury a citací

[1] Fuchs, Pavel, *Využití spolehlivosti v provozní praxi*, TUL, Liberec, listopad 2002

[2] Fuchs P., Vališ D., *Metody analýzy a řízení rizika*, TUL, Liberec, 2004

[3] Dhillon, B.S., *Engineering Maintenance*, Boca Raton: CRC Press 2002

[4] Lacko, Branislav, Vysoké učení technické v Brně, FSI, 2003, URL: http://honor.fi.muni.cz/tsw/2003/076.pdf

[5] Kolektiv autorů - Fuchs P., Vališ D., ad. Technická univerzita v Liberci, URL: www.rss.tul.cz/index.php?page=studium/predmet&zkratka=rjs

[6] Balakrishnan, N., Rao, C. R., Handbook of statistics, Vol. 20: Elsevier 2001

[7] Fuchs P., Vališ D., Chudoba J., Kamenický J., Zajíček J., *Bezporuchovost a životnost - Techniky analýzy bezporuchovosti*, skripta Technické univerzity v Liberci

[8] ČSN IEC 50 191 (01 0102): Medzinárodný elektrotechnický slovník, Kapitola 191: Spoľahlivosť a akosť služeb. Bratislava, Federálný úrad pro normalizacii a merení, září 1993, 168 stran

[9] HongzhouWang and Hoang Pham: Reliability and Optimal Maintenance, London: Springer-Verlag Limited, 2006. Zde Chapter 11

[10] Hoang Pham, Handbook of Reliability Engineering, London: Springer Verlag Limited 2003. Zde Chapter 11.

[11] Dalal S.R., Lyu M.R., Mallows C.L., Software Reliability, Bellcore

[12] Kan Stephen H., Metrics and Models in software quality Engineering, (2nd Edition) Addison-Wesley 2002

[13] Xie Min, Software Reliability Modelling, World Scientific, Singapore, 1991

[14] Skařupa Jiří, Kreativita a inovační myšlení v konstruování, VŠB Ostrava 2007, URL: http://www.fs.vsb.cz/euprojekty/415/kreativita-a-inovace.pdf

[15] Koucký Miroslav, ITEM ToolKit Sofware - výpočetní modul pro markovskou analýzu, Liberec 2008

[16] ČSN IEC 60300: Management spolehlivosti, Český normalizační institut, 2002

[17] ČSN IEC 60812: Techniky analýzy bezporuchovosti systémů – Postup analýzy způsobů a důsledků poruch (FMEA), Český normalizační institut, 2007

[18] ČSN EN 61025 : Analýza stromu poruchových stavů (FTA), Český normalizační institut, 2007

[19] ČSN EN 61078: Techniky analýzy spolehlivosti - Blokový diagram bezporuchovosti a booleovské metody, Český normalizační institut, 2007

[20] ČSN EN 61165: Použití Markovových technik, Český normalizační institut, 2007

[21] ČSN ISO/IEC 12207: Informační technologie – Procesy v životním cyklu softwaru, Český normalizační institut, 1997

[22] Software reliability, Pham Hoang, Springer 2000

Seznam obrázků